CLAIMS

What is claimed is:

1. A method to efficiently design and implement a matched instruction set processor system, including:

decomposing the matched instruction set processor system into interconnected design vectors; and

analyzing and mapping the interconnected design vectors into specific hardware and software elements.

2. The method of claim 1, wherein decomposing the matched instruction set processor into interconnected design vectors includes:

performing a concurrency analysis to determine concurrency of execution of actors.

3. The method of claim 2, wherein performing a concurrency analysis includes:

analyzing an order of invocation of the actors to determine an invocation type, wherein the invocation type is one of coterminous invocation, sequential invocation, and overlapping pipelined invocation.

4. The method of claim 1, wherein decomposing the matched instruction set processor into interconnected design vectors includes:

performing a concurrency analysis to identify an invocation period, the invocation period representing a time duration between two successive invocations.

5. The method of claim 1, wherein decomposing the matched instruction set processor into interconnected design vectors includes:

performing a concurrency analysis to identify a maximum allowable response time, the maximum allowable response time representing a maximum time duration during which processing should be completed.

6. The method of claim 1, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing model atomization to convert an actor into application specific code.

7. The method of claim 6, wherein performing model atomization to convert an actor into application specific code includes:

removing parameters by converting the parameters to private data constants inside the code; and

removing data polymorphism and domain polymorphism.

8. The method of claim 1, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing functional vector creation to put an actor into design vector format.

9. The method of claim 8, wherein performing functional vector creation to put an actor into design vector format includes:

creating a design vector;

putting a fire() method into the design vector;

creating a header data of the design vector;

generating the trailer data of the design vector; and

creating binding methods for the design vector.

10. The method of claim 1, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing a model test and verification.

11. The method of claim 1, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing a functional vector extraction to extract source code for functional vectors from encapsulated actors.

12. The method of claim 1, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing an interconnect vector extraction to extract source code for interconnect vectors from encapsulated actors.

13. The method of claim 1, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing a Conjugate Virtual Machine (CVM) generation to extract CVM instructions.

14. The method of claim 1, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing a stand-alone testing of extracted design vectors.

15. A machine-readable medium comprising instructions which, when executed by a machine, cause the machine to perform operations comprising:

decomposing the matched instruction set processor system into interconnected design vectors; and

analyzing and mapping the interconnected design vectors into specific hardware and software elements.

16. The machine-readable medium of claim 15, wherein decomposing the matched instruction set processor into interconnected design vectors includes:

performing a concurrency analysis to determine an execution order of actors.

17. The machine-readable medium of claim 15, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing model atomization to convert an actor into application specific code.

18. The machine-readable medium of claim 15, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing functional vector creation to put an actor into design vector format.

19. The machine-readable medium of claim 15, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing a model test and verification.

20. The machine-readable medium of claim 15, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing a functional vector extraction to extract source code for functional vectors from encapsulated actors.

21. The machine-readable medium of claim 15, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing an interconnect vector extraction to extract source code for interconnect vectors from encapsulated actors.

22. The machine-readable medium of claim 15, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing a Conjugate Virtual Machine (CVM) generation to extract CVM instructions.

23. The machine-readable medium of claim 15, wherein decomposing the matched instruction set processor system into interconnected design vectors includes:

performing a stand-alone testing of extracted design vectors.

24. An architectural modeling apparatus to decomposing a matched instruction set processor system into interconnected design vectors, comprising:

a concurrency analyzer to determine an execution order of actors; and

a model atomizer to convert the actors into application specific code.

25. The architectural modeling apparatus of claim 24, further comprises:

a functional vector creator to put the actors into design vector format.

26. The architectural modeling apparatus of claim 24, further comprises:

a testing unit to verify a model.

27. The architectural modeling apparatus of claim 24, further comprises:

a functional vector extractor to extract source code for functional vectors from encapsulated actors.

28. The architectural modeling apparatus of claim 24, further comprises:

an interconnect vector extractor to extract source code for interconnect vectors from encapsulated actors.

29. The architectural modeling apparatus of claim 24, further comprises:

a Conjugate Virtual Machine (CVM) generator to extract CVM instructions.